Full Stack Web Developer Nanodegree Syllabus

Build Complex Web Applications

Before You Start

Thank you for your interest in the Full Stack Web Developer Nanodegree!

In order to succeed in this program, we recommend having experience programing in HTML, CSS and programming languages like Python and Javascript. If you've never written code before, we recommend starting with the Introduction to Programming Nanodegree Program, which will prepare you for this and other career-focused Nanodegree programs.

Prerequisites:
You will need to be able to communicate fluently and professionally in written and spoken English.

To enroll, you should also have experience in the following courses or skills:

Programming with Python or another object-oriented programming language
Programming with JavaScript
Data Structures including Lists, Arrays, Dictionaries
Git/GitHub
Introduction to HTML

Educational Objectives:
Students will learn about building out the infrastructure that powers and supports the many web, desktop, mobile and integrated applications in the world.

Length of Program*: 160 Hours
Textbooks required: None
Instructional Tools Available: Video lectures, Mentors, Forums

*The length is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 10 hours per week working through the program, you should finish in 16 weeks, so approximately 4 months. Actual hours may vary.

Part 1. Developers' Tools

Brush up your knowledge of essential developers' tools such as the Unix shell, Git, and Github; then apply your skills to investigate HTTP, the Web's fundamental protocol.

Lessons: Shell Workshop

Lesson Title	Learning Outcomes
Shell Workshop	➔ The Unix shell is a powerful tool for developers of all sorts. Get a quick introduction to the basics of using it on your computer.

Lessons: Git & Github

Lesson Title	Learning Outcomes
Purpose & Terminology	➔ Learn about the benefits of version control and install the version control tool Git.
Create a Git Repo	➔ Create a new Git repository for your code.
Review a Repo's History	➔ Review an existing Git repository's history of commits — the changes that have been made to the project.
Add Commits to a Repo	➔ A repository is nothing without commits. In this lesson, you'll learn how to make commits, write descriptive commit messages, and verify the changes you're about to save to the repository.
Tagging, Branching, and Merging	➔ Being able to work on your project in isolation from other changes will multiply your productivity. You'll learn how to do this isolated development with Git's branches.
Undoing Changes	➔ Help! Disaster has struck! You don't have to worry, though, because your project is tracked in version control! You'll learn how to undo and modify changes that have been saved to the repository
Working with Remotes	➔ Create remote repositories on GitHub and send changes to the remote repository
Working on Another Developer's Repository	➔ Fork another developer's project and learn how to contribute to a public project
Staying In Sync With a Remote Repository	➔ Send suggested changes to another developer by using pull requests and use the powerful `git rebase` command to squash commits together

Lessons: HTTP & Web Servers

Lesson Title	Learning Outcomes
Requests & Responses	➔ Examine HTTP requests and responses by experimenting directly with a web server, interacting with it by hand.
The Web from Python	➔ Build up your knowledge of HTTP by writing servers and clients in Python that speak HTTP.
HTTP in the Real World	➔ Examine a number of practical HTTP features that go beyond basic requests and responses.

Lessons: Networking for Developers

Lesson Title	Learning Outcomes
From Ping to HTML	➔ Examine HTTP requests and responses by experimenting directly with a web server, interacting with it by hand.
The Web from Python	➔ Build up your knowledge of HTTP by writing servers and clients in Python that speak HTTP.
HTTP in the Real World	➔ Examine a number of practical HTTP features that go beyond basic requests and responses.

Part 2. Databases with SQL and Python

Master SQL databases and build multi-user web applications using the Flask framework, SQLAlchemy, and authentication providers such as Google and Facebook.

Lessons: Intro to Relational Databases

Lesson Title	Learning Outcomes
Data and Tables	➔ Use the table structure of databases to organize data ➔ Use types and keys to more accurately model your data
Elements of SQL	➔ Use the `select` statement to retrieve data from tables ➔ Use the `insert` statement to add data to tables ➔ Combine SQL tables using joins and aggregations to create powerful queries
Python DB-API	➔ Interact with a database from Python code ➔ Connect a Python web application to an SQL database

	➔ Discover and fix security problems with database-backed apps
Deeper into SQL	➔ Create tables using normalized forms ➔ Use keys to express relationships between tables ➔ Write reusable views to quickly and efficiently retrieve data

Project: Logs Analysis

In this project, you'll analyze data from a web service's logs, practicing your command-line and database skills, particularly with a focus on building advanced SQL queries.

Part 3: Servers, Authorization, and CRUD

Learn the CRUD pattern (Create, Read, Update, Delete) and how it relates to RESTful architectures and to the operations of a database-backed web service. Learn the difference between authentication and authorization and some best practices in developing a login system.

Lessons: Full Stack Foundations

Lesson Title	Learning Outcomes
Working with CRUD	➔ Model database entries in Python ➔ Write server code to create, read, update and delete database entries interactively.
Making a Web Server	➔ Configure a web server to handle requests using HTTP ➔ Allow a web server to read and update data based on HTTP request input
Developing with Frameworks	➔ Build a functioning web application using the lightweight Flask framework ➔ Respond to HTTP requests with JSON data
Iterative Development	➔ Plan the design of a complex web application

Lessons: Authentication and Authorization

Lesson Title	Learning Outcomes
Authentication vs Authorization	➔ Secure your application by verifying users' identities ➔ Control application authorization based on user roles and login state ➔ Use third-party systems to authenticate users
Creating Google Sign-in	➔ Implement user authentication using Google's OAuth 2.0 tools
Local Permission System	➔ Store user data in an application database ➔ Manage user authorization from stored user data
Adding Facebook and Other Providers	➔ Implement other authentication providers in a web app

Lessons: RESTful APIs

Lesson Title	Learning Outcomes
What's and Why's of APIs	➔ Examine API terminology, techniques, and the REST concept.
Accessing Published APIs	➔ Send requests to remote APIs. Use published documentation to understand and apply those APIs correctly.
Creating Your Own APIs	➔ Apply the Flask framework to create APIs in Python code.
Securing Your API	➔ Use token-based authentication and OAuth to protect API endpoints.
Writing Developer-Friendly APIs	➔ Improve your documentation skills and use API versioning to help developers use your API correctly.

Project: Build an Item Catalog

In this project, you will develop an application that provides a list of items within a variety of categories as well as provide a user registration and authentication system. Registered users will have the ability to post, edit and delete their own items.

Part 4: Deploying to Linux Servers

Lessons: Configuring Linux Web Servers

Lesson Title	Learning Outcomes
Intro to Linux	➔ Explore the historical roots of Linux and some common Linux distributions ➔ Launch the Ubuntu operating system in a virtual machine on your own computer
Linux Security	➔ Control authorization on a Linux system using super user privileges ➔ Install additional software packages to a Linux system ➔ Manage Linux users and user permissions ➔ Protect a Linux system with a universal firewall
Web Application Servers	➔ Install an Apache web application server on a Linux system

Project: Linux Server Configuration

In this project, you will take a baseline installation of a Linux distribution on a virtual machine and prepare it to host your web applications, to include installing updates, securing it from a number of attack vectors, and installing and configuring web and database servers.

Extracurricular Material

Lesson Content: Web Accessibility

Lesson Title	Learning Outcomes
Accessibility Overview	➔ Explore the diversity of different users experience with websites and applications. Learn about using screen readers practically and recognize the challenge of building web experiences for all users.
Focus	➔ Learn how important focus is to maintain an accessible site. Maintain focus using the Tabindex, Keyboard Design Patterns, and Offscreen Content.
Semantics Basics	➔ Dive into the differences between visual UI and semantically designed accessible UI. Add semantic elements to HTML to create a user interface that works for everyone.

Navigating Content	➔ Implement effective semantic navigation using headings, link text and landmarks.
ARIA	➔ Sometimes an HTML element may not have a role or value assigned semantically. In this lesson, you'll use ARIA attributes to provide context for screen readers.
Style	➔ Incorporate CSS styling into your accessible web design and use accessible color schemes to improve accessibility.

Lessons: JavaScript Design Patterns

Lesson Title	Learning Outcomes
Changing Expectations	➔ React to changing product specifications and developer expectations ➔ Explore the Model-View-Controller design pattern ➔ Analyze an existing application for MVC structure
Refactoring with Separation of Concerns	➔ Write code with discrete areas of responsibility in an MVC application ➔ Refactor an existing application to make use of modern code design practices
Using an Organization Library	➔ Build a reactive front end application using an organization library, knockout.js ➔ Implement knockout models and observable elements in an application
Learning a New Codebase	➔ Use proven strategies to adapt to a new and unfamiliar codebase

Lessons: Intro to AJAX

Lesson Title	Learning Outcomes
Requests and APIs	➔ Connect to external web APIs to power asynchronous browser updates
Building the Move Planner App	➔ Use the jQuery Javascript library to build AJAX requests and handle API responses ➔ Handle error responses with AJAX